

NASA Contractor Report 4428

1N-65  
72172  
p-35

# Study on Advanced Information Processing System

Kang G. Shin and Jyh-Charn Liu

GRANT NAG1-981  
FEBRUARY 1992

(NASA-CR-4428) STUDY ON ADVANCED  
INFORMATION PROCESSING SYSTEM Final Report  
(Michigan Univ.) 35 p CSCL 12A

N92-19536

H1/65 0072172  
Unclas

**NASA**



NASA Contractor Report 4428

# Study on Advanced Information Processing System

Kang G. Shin and Jyh-Charn Liu  
*The University of Michigan*  
*Ann Arbor, Michigan*

Prepared for  
Langley Research Center  
under Grant NAG1-981



National Aeronautics and  
Space Administration

Office of Management

Scientific and Technical  
Information Program

1992

# 1 INTRODUCTION

Embedded computing systems are becoming essential to many critical applications, such as flight control and life support systems. To meet the requirement of uninterruptible service during each mission, such a system is often built with multiple *computing channels* so that faults can be masked during normal operation. Each of these redundant channels is physically independent of others and is composed of a complete set of processors, memory, and other control circuits, so that each fault-free channel can accomplish the functions assigned to the original set of redundant channels. An  $N$  modular redundant (NMR) system can mask up to  $\lfloor \frac{N-1}{2} \rfloor$  faults by tightly synchronizing, and voting on, the operations of  $N$  channels. A channel is said to have error(s) when (physical) faults within the channel manifest themselves, generating outputs different from those generated by a fault-free channel.

One key design consideration of NMR systems is the frequency of voting on channels' outputs. Increasing the voting frequency can improve fault detection capability but may degrade system performance. Different designs of voters can be found in the Fault-Tolerant Multiprocessor (FTMP) [1] and the Fault-Tolerant Processor (FTP) [2], both of which were intended for life-critical real-time applications like flight control for commercial airplanes. In FTMP every common memory access requires voting<sup>1</sup>, whereas in FTP votes are taken only on certain data. Thus, by its less intrusive voting, FTP can provide better performance than FTMP. The reliability and fault recovery problems of FTP are explored through a case study for an unmanned reusable launching system called the Advanced Launch System (ALS). ALS is a typical application of a class of architecture called the *Advanced Information Processing Systems* (AIPS), which is currently being developed by Charles Stark Draper Laboratory (CSDL) under the support of the National Aeronautics and Space Administration (NASA).

FTP voter design imposes few constraints on the architecture but is relatively slow,

---

<sup>1</sup>Cache memory accesses are not voted on, though.

i.e., it takes several steps to vote on a message/data, and each step takes approximately 5  $\mu$ seconds, when compared to the instruction cycles of contemporary microprocessors. This in turn implies that FTP's channel outputs should not be voted on frequently. As embedded systems are becoming increasingly complex, one must carefully investigate the dynamics of system failure for life-critical applications with long mission times.

The first important problem to be addressed in this report is the probability of system failure due to nearly-coincident faults in FTP. By developing a realistic system model, we shall show this probability to be negligible. Then, we analyze the probability of resource exhaustion<sup>2</sup> for applications with long mission times. A serious drawback of low-speed (slack) voters is that when channels have large main memory, it is very time-consuming to re-align all channels with a slack voter into an identical state.

To alleviate the difficulty associated with memory re-alignment, a monitoring technique using signature analysis was proposed in [3]. In this method, main memory is decomposed into signature pages, and memory accesses to each page are encoded into a signature which is then stored in an independent signature memory. A fault is thus detectable only when a faulty word is accessed. Upon detection of a fault in main memory, only those pages with different signatures need to be re-aligned. The signature analysis cannot completely overcome the memory re-alignment problem, because even though a massive redundant system may have congruent inputs, errors caused by random permanent/transient faults that occur in memory cells cannot be detected by this technique. Thus, in the worst case, the whole main memory must be realigned when such faults occur.

Channel failure rate has the most pronounced effects on system reliability, because it (i) determines when a resource exhaustion occurs, and (ii) affects the process of fault detection, fault location, and reconfiguration. When channel failure rate is high, the success of a mission will be greatly affected by the quality of fault handling processes. On the other hand, when channel failure rate is low, rare occurrences of faults will lower the demand on

---

<sup>2</sup>due to loss of resources as a result of failures

system resources (e.g., hardware, computing time, and software routines) for fault handling. System design would thus be greatly simplified if the channel failure rate can be effectively reduced.

As a solution to both the reliability enhancement and memory re-alignment problems, we propose to use *channel error maskers* (CEMs). The main motivation behind this proposal is to make channels more reliable by masking channel faults with CEMs. When the reliability of each channel is improved, the need of memory re-alignment can be reduced significantly. It is shown that the reliability of ALS is dramatically improved when the CEMs for main memory are implemented by common single error correction/double error detection (SEC/DED) codes. Furthermore, using CEMs can speed up the memory re-alignment process substantially, because only those faulty words uncovered by CEMs need to be recovered by the voter. Two different schemes, called **Scheme\_1** and **Scheme\_2**, respectively, are developed for the re-alignment of main memory. In **Scheme\_1**, main memory is decomposed into recovery pages, and a page is re-aligned only when it cannot be recovered by CEMs. An optimization technique is developed to find the optimal page size for **Scheme\_1**. In **Scheme\_2**, addresses of faulty memory words are recorded, and only those recorded faulty words need to be re-aligned.

In order to assess the FTP's capability for the ALS mission, the basic operational principles of FTP are first introduced in Section 2. In Section 3, we develop a reliability model which is then used to evaluate the FTP's reliability for ALS. CEMs are then applied to solve the memory re-alignment problem in Section 4. The report concludes with a few remarks in Section 5.

## 2 FTP Architecture

The architecture of FTP, and the memory access model for the analysis of multiple channel faults, are introduced in this section. An FTP channel may have one or two processors. When two processors are built into each channel, one processor is dedicated to

computation functions and the other to I/O functions. The two processors in a channel communicate with each other by writing and reading messages in a shared memory. There are interval timers and watchdog timers in each channel for task scheduling and time-out interrupts.

FTP can provide high performance, and its architecture can be in the form of simplex, duplex, triplex (TMR), or quadruplex (QMR). In a redundant FTP system, only clocks in the different channels of FTP are tightly synchronized, and thus, fault-free channels can execute identical instructions in lock-step. Channels communicate with each other via a network formed by the *communicators* in redundant channels.

A block diagram of the communicator network in a QMR FTP is given in Fig. 1 where a communicator is composed of a set of registers, a transmitter (single input,  $N$  outputs), interstage (single input,  $N$  outputs), and a receiver ( $N$  inputs, single output). There are four channels, called channel  $A, B, C$  and  $D$ , respectively. The set of registers  $X_V, X_R, X_E$ , and  $X_Y$  in channel  $Y, Y \in \{A, B, C, D\}$ , store inputs and outputs of channel  $Y$  to/from the communicator network.

Logically, channels exchange data by reading/writing data from/to the set of registers in the communicator. Data communications between channels are classified as *voted data-exchange*, and *simplex data-exchange*. FTP design emphasizes the concept of *source congruency*: for all types of data-exchanges, all correctly operating channels will eventually receive identical copies of data. A voted data-exchange allows channels to compare their outputs and mask any error whenever possible. A voted data-exchange is accomplished by writing a value to  $X_V$ , and then reading the voted result from  $X_R$ . Register  $X_E$  in each channel records any discrepancy between its  $X_V$  and  $X_R$  values. The actual steps in a voted data-exchange are that (1) every channel sends a message to its transmitter which will then relay the message to its own interstage, and (2) through the fully-connected network from  $N$  interstages to  $N$  receivers, the receiver in every channel gets a voted message and stores it in  $X_R$ .

A simplex data-exchange can be used by a channel to broadcast messages to the other channels. For example, if a message needs to be transferred from channel  $A$  to the others, all channels execute an instruction “write message  $\Phi$  to  $X_A$ ”. When the instruction “write  $\Phi$  to  $X_A$ ” is executed by channel  $A$ ,  $\Phi$  will be broadcast via the transmitters to all interstages. In the meantime, the pseudo-messages  $\Phi$  sent by channels  $B, C$  and  $D$  are discarded by the communicator network. After all interstages receive replicated copies of  $\Phi$ ,  $\Phi$  is broadcast on the interstage network, and every receiver will have the voted  $\Phi$  stored in  $X_R$ ’s. Through such an exchange process, data congruency is guaranteed for both voted and simplex data-exchanges.

### 3 Reliability Analysis

To justify the use of a single fault model, Section 3.1 presents a preliminary examination of system failures due to nearly-coincident faults when the system uses memory coding and segmentation. Then, using the single fault model, the probability of system failure due to exhaustion-of-parts, and the reliability impact of CEMs on the ALS will be analyzed in Section 3.2.

#### 3.1 Probability of Nearly-Coincident Faults

A complete reliability model of FTP must include both sequential and nearly-coincident faults. However, incorporation of nearly-coincident faults into the reliability model will make the analysis very complex, because it deals with a multivariate distribution. To alleviate the complexity, calculation of the probability of nearly-coincident faults is separated from that of the probability of system crash caused by resource depletion.

Since a channel’s operation depends heavily on its access to main memory, a memory access model needs to be developed for the evaluation of nearly-coincident faults. The existence of memory access locality has been the key to the modeling of memory access

behavior. That is, once a program starts to access a specific memory area, it tends to access the area continuously for a certain period. Thus, although memory cells are physically identical, different parts of main memory must be distinguished when they have different logical uses.

Using memory access locality, a program's memory access behavior can be modeled by an *active agent* visiting main memory and forming access sets [4]. An *access set* is defined as a memory area that is continuously accessed for a certain period of time during each visit. We further assume that (1) locations of all access sets in the system do not change, (2) the number of access sets in the system is fixed, (3) all access sets have the same size  $u$  and are disjoint with one another, and (4) memory segmentation and single-error-correction and double-error-detection (SEC/DED) codes are implemented in the memory, so that latent faults are covered by a background scrubbing process and the SEC/DED codes when the memory is not accessed. A faulty memory segment is replaced at the end of the current visit of the active agent. Optimization of the memory segment replacement procedure is discussed in Section 4. Based on these assumptions, an access set can be used to denote a set of "physical" memory cells in a certain area.

There are  $m$  access sets  $\{AS^1, AS^2, \dots, AS^m\}$  in main memory. Let  $V_j^i$  denote the event of the agent's  $j$ -th visit to  $AS^i$  and  $M_j^i \in \mathbb{R}^+ \cup \{0\}$  be the  $V_j^i$ 's lifetime.  $M_j^i$ 's are assumed to be independent and identically distributed (i.i.d.) random variables, and the agent's present and future visits are independent of its past visits. Based on this memory access model, a nearly-coincident fault occurs when more than one agent (i.e., channel processor) either become faulty or visit faulty access sets during one inter-voting period.

To calculate the probability of nearly-coincident faults, it is assumed that memory is free of faults initially and is not re-aligned when the voter can mask faulty channel outputs. Voters are assumed to be the only fault detection/masking mechanism in the system. Once the processor enters an access set, any of the faults in the processor, voter, or the access set will cause a faulty output on the channel. Note that faults in one access set do not



propagate to another access set. For convenience of presentation, all processor and voter faults are classified as access set faults. Since the component failure rate is very low as compared to memory access times, a large number of access sets will be visited between two successive fault occurrences, and no new fault is likely to occur in an access set during a visit to the access set. Faults occur independently according to a Poisson process.

A system with  $m$  access sets is said to be in *state*  $i$  when the agent is in  $AS^i$ . Let  $S_k^i$  be the time  $V_k^i$  begins — the agent's  $k$ -th visit to access set  $AS^i$  — and let  $N^i(t) = \sup \{k \mid S_k^i \leq t\}$ ,  $N^i(t) \in \mathbb{I}^+ \cup \{0\}$ ,  $\forall i, t$ . For a given time interval  $[0, t]$ ,  $t > 0$ , the total number of visits made by the agent to access sets is  $N(t) \equiv \sum_{k=1}^m N^k(t)$ . One or more access sets may have been visited before the channels vote on their outputs. Let the random variable  $X_i^j \in \mathbb{I}^+ \cup \{0\}$  denote the number of faults occurred in  $AS^j$  during the agent's  $i$ -th visit (to some access set). Since faults occur uniformly within memory and  $X_i^j$ 's are assumed to be i.i.d.,  $X_i^j$ 's will be represented by a single random variable  $X$ . Thus, at any time instant, the agent's decision on which access set to visit makes no difference. Let  $Y_i = T_i - T_{i-1}$  where  $T_j$  is the time of the  $j$ -th voting on channel outputs. When  $Y_i$ 's are i.i.d., they can be represented by a single random variable  $Y$ .

Assuming that the agent has visited  $\ell$  access sets during  $[0, T_{j-1})$ , at time  $T_{j-1}^+$  there are  $\ell X$  faults in  $AS^i$ . In an NMR system, let  $P_c(Y_i)$  be the probability of a channel generating a faulty output during the time interval  $[T_{j-1}, T_j)$ . During  $[0, T_j)$ , the total probability of system crash due to nearly-coincident faults becomes

$$P_N(T_j) = \sum_{k=1}^j \sum_{i=\lceil \frac{N}{2} \rceil}^N \binom{N}{i} Pr(\text{no failure before } T_{k-1}) (P_c(Y_k))^i (1 - P_c(Y_k))^{N-i} \quad (3.1)$$

$$\leq \sum_{k=1}^j \sum_{i=\lceil \frac{N}{2} \rceil}^N \binom{N}{i} (P_c(Y_k))^i (1 - P_c(Y_k))^{N-i}. \quad (3.2)$$

To evaluate  $P_c(Y_i)$ , within  $[T_{j-1}, T_j)$  the probability of a faulty output generated by a channel is

$$P_c(Y_j | w \text{ visits between two successive votings}) = 1 - (P(X = 0))^w$$

$$\begin{aligned}
&= 1 - e^{-wu\lambda Y} \\
&\approx wu\lambda Y.
\end{aligned} \tag{3.3}$$

where  $u$  and  $\lambda$  are the size of access set and the failure rate of a memory word, respectively. When  $w = 1$ , i.e., channels vote on their results after accessing each access set, Eq. (3.2) can be simplified as

$$P_N(T_j) < j \sum_{i=\lceil \frac{N}{2} \rceil}^N \binom{N}{i} (u\lambda Y)^i (1 - u\lambda Y)^{N-i}. \tag{3.4}$$

### 3.2 Analysis of ALS

In this subsection, the probability of system crash due to nearly-coincident faults, and the effectiveness of CEMs are discussed using the ALS mission scenario. The ALS will first sit on the launching pad for a week, and will then be in the boost phase for 10 minutes. Any approval for launch requires the system to have fault masking capability. The system must have 0.95 probability of availability, 0.98 probability of mission success, and less than  $10^{-5}$  system unreliability at the end of mission. Since information on the maintenance schedule and the requirement for mission success are not available, we will focus on system reliability and the probability of system possessing fault masking capability before launch.

The parameters necessary to estimate the reliability of ALS are derived from the results of the Entry Research Vehicle (ERV) study. Permanent failure rates of the processors (including control circuits) and the interstage are predicted to be  $\lambda_p = 8.91 \times 10^{-6}$ /hour, and  $\lambda_i = 1 \times 10^{-6}$ /hour, respectively [5]. Permanent failure rates of  $64K \times 4$  RAM chips and  $128K \times 8$  ROM chips are predicted to be  $6 \times 10^{-6}$ /hour, and  $2.8 \times 10^{-6}$ /hour, respectively. A redundant FTP equipped with 1M bytes of ROM and RAM in each channel is considered as an example ALS controller. Thus, the main memory needs 32 (8) RAM (ROM) chips, and the total failure rate of RAM (ROM) is  $\lambda_m = 192 \times 10^{-6}$ /hour ( $\lambda_o = 20.8 \times 10^{-6}$ /hour). Note that the above failure rates have been adjusted by the environment and quality factors,  $\Pi_e$  and  $\Pi_q$ , i.e.,  $\lambda_x \leftarrow \Pi \lambda_x$ , where  $x \in \{p, i, o, m\}$ , and  $\Pi = \Pi_e \Pi_q$ . Since  $\Pi_q = 0.5$  and

$\Pi_e = 3$  in the ERV study, the actual component failure rates are  $\lambda_p = 5.94 \times 10^{-6}/\text{hour}$ ,  $\lambda_o = 13.86 \times 10^{-6}/\text{hour}$ ,  $\lambda_i = 1 \times 10^{-6}/\text{hour}$ , and  $\lambda_m = 128 \times 10^{-6}/\text{hour}$ , respectively. With these parameter values, one can see that in the ALS, 96% of the channel faults are caused by main memory faults. This can be broken down to 86.8% of the faults due to RAM and 9.2% due to ROM.

The system cycle of FTP is 40 msec, within which all the essential control functions, including fault recovery processes, must be completed for the system to function acceptably. It takes about 11  $\mu\text{seconds}$  to vote on one memory word — the processor reads a memory word, votes on it, reads the result back from the voter, and then writes the voted word back to the memory. Because of the relatively low system failure rate, and the frequent memory scrubbing, it is reasonable to assume that the system is free of latent faults.

Note that when a fault occurs in the access set that is currently being visited by the agent, the fault cannot be detected/corrected by the scrubbing process, because the scrubbing process possesses the lowest priority. In the FTP, computing channels vote on their outputs at least every 40 mseconds, i.e.,  $T_i - T_{i-1} < 40$  mseconds. Thus, given that no fault occurs before  $T_{i-1}$ , and  $\frac{1}{T_i - T_{i-1}} \ll \lambda_a$ , where  $\lambda_a$  is the failure rate of an access set (including processor, interstage, memory and the access set itself), the probability of system crash due to nearly-coincident faults in an NMR system during  $Y_i = [T_i, T_{i-1})$  is

$$\begin{aligned} P_c(Y_i) &= \sum_{j=\lceil \frac{N}{2} \rceil}^N \binom{N}{j} e^{-(N-j)\lambda_a T_i} (e^{-\lambda_a T_{i-1}} \lambda_a Y_i)^j \\ &\approx \sum_{j=\lceil \frac{N}{2} \rceil}^N \binom{N}{j} e^{-N\lambda_a T_{i-1}} (\lambda_a Y)^j. \end{aligned} \quad (3.5)$$

Within  $[0, t)$ , the total probability of system crash due to two channel errors is

$$P_{crash}(t) < \frac{t}{Y} \binom{4}{2} (Y \times \lambda_a)^2 + O(h) \quad (3.6)$$

$$< t6Y\lambda_a^2. \quad (3.7)$$

In Eq. (3.7), the probability that the system does not crash before  $t$  is not considered, and  $O(h)$  is the probability of 3 or more channels becoming faulty simultaneously. The



probability of system crash due to nearly-coincident faults in the FTP is plotted in Fig. 2. This probability is shown to be very small even when the the size of access set is very large.

After evaluating the probability of nearly-coincident faults, one can develop a continuous-time Markov model for the reliability analysis of a QMR system due to resource exhaustion. As shown in Fig. 3, states A, B, C, D and E are used to denote the conditions where the system has four, three, two, one fault-free channels, and system crash, respectively. The model can be modified for a TMR system with state A removed. In this model,  $\lambda_t$  ( $\lambda_h$ ) is the failure rate of transient (hard) faults,  $c$  is the recovery coverage of transient faults, and  $c_d$  is the reconfiguration coverage of a duplex configuration. Assuming that a channel will be retired if any of its components becomes faulty, the total failure rate of a channel is  $\lambda_c = \lambda_p + \lambda_m + \lambda_o + \lambda_i$ . (See Appendix A for definitions of  $\lambda$ 's.)

A similar, but more complicated, FTP reliability model has been developed by CSDL [5]. In the CSDL's model, every component failure is considered to be an independent event, and the system reconfiguration time is treated as a random variable with an exponential distribution. Our model differs from CSDL's in that (1) system states are defined by the number of fault-free channels, (2) different component failures in one channel are aggregated into one single event, because when component failures are memoryless, and reconfiguration rates for different component failures are the same, the channel failure rate is the sum of component failure rates, and (3) system reconfiguration is considered to be done instantaneously, because it is usually done in one system cycle 40 mseconds, or 9000/hour, which is extremely fast relative to faults' inter-arrival times.

Next, we want to evaluate the effectiveness of CEMs. A channel with embedded CEMs will be retired if CEMs become faulty. Thus, the channel failure rate becomes  $\lambda_c = \lambda_p^C + \lambda_i^C + \lambda_o^C + \lambda_m^C + (1 - c_p)\lambda_p + (1 - c_i)\lambda_i + (1 - c_o)\lambda_o + (1 - c_m)\lambda_m$ , where  $c_p, c_i, c_o$ , and  $c_m$  ( $\lambda_p^C, \lambda_i^C, \lambda_o^C, \lambda_m^C$ ) are the coverage (failure rate) of CEMs for processors, interstage, ROM and RAM, respectively.

As mentioned in the beginning of this subsection, 96% of channel failures are due to

main memory failures. Thus, adding CEMs to main memory can dramatically reduce the channel failure rate. On the other hand, CEMs could be designed for processors (and control circuits), but this is more difficult and has little impact on system reliability, since only 4% of channel failures result from this portion of hardware. Consequently, the design of CEMs for processors will not be considered any further. Note that CEMs would be inefficient if they could not achieve high fault coverage during the mission. Assuming that CEMs for memory can correct  $w$  bit-errors out of an  $n$ -bit word, and faults in memory bits are independent of each other, one can derive the coverage of CEMs at time  $t$  as:

$$c_m(c_o) = \frac{\sum_{i=1}^w \binom{n}{i} (1 - e^{-\frac{1}{n}\lambda t})^i e^{-\frac{n-i}{n}\lambda t}}{1 - e^{-\lambda t}}, \quad (3.8)$$

where  $\lambda$  is the failure rate of a memory word. For the FTP example, if we use 7 extra bits to encode a 32-bit data word by SEC/DED codes, we get  $\lambda = \frac{39}{32} \times 10^{-9}$ /hour-word, and  $c_m \approx c_o \approx 1 - 2 \times 10^{-7}$  at  $t = 200$  hours. However, when multiple-bit chips are used, other coding schemes should be employed [6] to provide high fault coverage. Since the implementation of CEMs for main memory is straightforward with standard commercial error controllers (e.g., 74ALS632B), they will not be discussed any further in this report.

Evaluation of the reliability of a redundant system with CEMs is very simple when the system has perfect reconfiguration capability. For example, consider two redundant systems with  $N$  and  $W$  computing channels, respectively. CEMs are embedded into the NMR system, denoted as NMR\_CEM, but no CEM is embedded into the WMR system. Let the channel failure rate of NMR\_CEM (WMR) be  $\lambda'_c$  ( $\lambda_c$ ), and  $N \leq W$ , then the probability of NMR\_CEM (WMR) crash before time  $t$  is  $P_N(t) = (1 - e^{-\lambda'_c t})^N$  ( $P_W(t) = (1 - e^{-\lambda t})^W$ ). When  $\lambda t \ll 1$  ( $\lambda' t \ll 1$ ),  $P_N(t) \approx (\lambda' t)^N$  ( $P_W(t) \approx (\lambda t)^W$ ). Thus, an NMR\_CEM is more reliable than a WMR system when  $\lambda' < \lambda \sqrt[N]{(\lambda t)^{W-N}}$ . Note, however, that a numerical method is usually called for when systems do not have perfect reconfiguration capability.

Using the component failure rates predicted by the ERV study, numerical solutions of the ALS reliability with and without CEMs are calculated by METASAN<sup>TM</sup> [7]. Let the

---

<sup>TM</sup>METASAN is a registered trademark of the Industrial Technology Institute.

failure rate of CEMs for memory be the same as that of an interstage, the coverage of transient faults be 1, and the coverage of duplex system be 0.9, i.e.,  $c_p = c_i = 0$ ,  $\lambda_m^C = \lambda_i$ ,  $c = 1$ , and  $c_d = 0.9$ . The probability of system crash on the launch pad for TMR and QMR systems with and without CEMs are plotted in Figs. 4 and 5. The two diagrams in Fig. 4 (5) show the reliability impacts of CEMs when  $\Pi = 0.1$  and  $\Pi = 1$ , respectively, where  $\Pi$  is an adjusting factor of channel failure rate. In Fig. 4, SEC/DED codes are embedded into RAM only, and in Fig. 5, SEC/DED codes are embedded into both ROM and RAM. Clearly, a TMR system with the entire memory (ROM and RAM) encoded is more reliable than a conventional QMR system even for very short missions and very low component failure rates. Furthermore, while the reliability improvement by changing from a conventional TMR to a QMR system is in the order of 10 to 100, when CEMs are embedded into main memory, the reliability improvement by upgrading a system from TMR\_CEM to QMR\_CEM is in the order of  $10^3$  to  $10^4$ .

The probability of FTP retaining fault masking capability for the ALS is examined next. As shown in Fig. 6, the probability that a conventional QMR system retaining fault masking capability decreases quickly with increases in  $\Pi$  (i.e., component failure rates) and launch waiting times. On the other hand, since channels in TMR\_CEM or QMR\_CEM are inherently reliable, the probability of launch approval increases substantially even for very long waiting times.

Finally, the total system reliability throughout the mission can be derived as follows. The system unreliability is the sum of the probability of system crash before launch, and the probability of system crash during the launch. Since the system cannot be launched unless the FTP has fault masking capability, we can calculate the probability of system crash during the boost period conditioned on that the FTP has fault masking capability. When the boost time is less than 20 minutes, the probability of system crash during the launch is lower than  $10^{-7}$  for systems without CEMs, and the figures are much lower for systems with CEMs. Thus, the probability of system crash during the on-pad waiting period is much higher than the probability of system crash during the boost phase.



## 4 Memory Re-alignment

Application of CEMs to the memory re-alignment problem is the subject of this section. In a conventional QMR system, the probability that the channels need to be re-aligned is  $P_r(t) = 1 - e^{-4\lambda_t t}$ , where  $\lambda_t$  is the transient failure rate of RAM. When  $\lambda_t = 128 \times 10^{-5}$ , we get  $P_r(200) \approx 0.64$ , implying that memory faults should be a serious concern to any system design.

Theoretically, when a transient fault occurs in memory, the fault can be corrected by memory re-alignment. However, since it is very time-consuming to re-align channel memories, and since it is difficult to discriminate permanent, intermittent, and transient faults in a limited amount of time, it is highly desirable to correct faults, if possible, by CEMs without using memory re-alignment. For example, when SEC/DED codes are embedded into main memory, the transient failure rate is reduced by a factor of  $2 \times 10^{-7}$ . Plugging the new failure rate into  $P_r(t)$ , we get  $P_r(200) \approx 2 \times 10^{-7}$ . Thus, for the ALS mission scenario, channels' main memory re-alignment is unlikely to be called for when CEMs are embedded into main memory.

In addition to dramatically reducing the need of memory re-alignment, the fault-masking capability of CEMs can be used to speed up the process of memory re-alignment substantially. Two schemes, called **Scheme\_1** and **Scheme\_2**, are developed for the re-alignment of main memory. In **Scheme\_1**, the entire memory space of  $W$  words is decomposed into  $K$  *recovery pages*,  $\Omega_1, \Omega_2, \dots, \Omega_K$ , where  $|\Omega_i| = \frac{W}{K}, i \leq K$ . When the system decides to start memory re-alignment, all channels scan through main memory page-by-page. After each page of different channels is scanned, channels have the scanned page re-aligned if any one of them is found to be faulty. The procedure is repeated until the entire memory system is completely scanned and/or re-aligned. When two pages have different sizes, we can repeatedly subtract 1 byte from the page of larger size, and add 1 byte to the other until the difference of their page size is less than, or equal to, 1. Thus, when  $\frac{W}{K}$  is not an integer, there is at most one byte difference among pages. Since the reliability difference

and re-alignment overhead caused by the one-byte difference in page size is negligible, it is assumed that  $K$  can always divide  $W$  without leaving a non-zero remainder.

In **Scheme\_2**, the entire memory is decomposed into  $\Omega_1$  and  $\Omega_2$ , where  $\Omega_1$  is a fault register of variable size, and  $\Omega_2$  is the rest of main memory. When main memory needs to be re-aligned, the CPU in each channel scans through its main memory and places addresses of faulty words into its fault register. After all channels complete their memory scan, they use simplex data-exchanges to broadcast addresses of faulty words, and then vote on each faulty word using the voted data-exchange.

Details of **Scheme\_1** and **Scheme\_2** are described in pseudo codes as follows.

**Scheme\_1(channel- $i$ )**

**begin**

Synchronize channels to start the re-alignment

$n = 1$ ;

**while** (  $n < K$  ) /\*scan recovery pages,  $K$  is the number of pages\*/

**do**

A="fault-free"; /\*The current page is assumed to be fault-free \*/

scan  $\Omega_n$ ;

**if** ( $\Omega_n$  faulty & cannot be corrected by CEMs) A="faulty";

**write** A to  $X_V$ ;

**if** ( $X_R = \text{"faulty"}$  or  $X_E \neq 0$  ) /\*at least one channel has a faulty page \*/

**do** /\*re-align  $\Omega_n$ \*/

j=1;

**while** ( $j \leq \frac{W}{K}$ )

**do write**  $\Omega_n(j)$  to  $X_V$ ;

**write**  $X_R$  to  $\Omega_n(j)$ ;

j=j+1;

**end\_do**

**end\_do**

**end\_do**

**end**

**Scheme\_2(channel- $i$ )**

**begin**

Synchronize channels to start the re-alignment

$j = 1;$

$k = 1;$

**while** (  $j < W$  ) /\*scan main memory,  $W$  is the total memory size\*/

**do**

**read**  $M(j);$  /\*read the  $j$ -th word\*/

**if** ( $M(j)$  faulty & cannot be corrected by CEMs)

**do**

**write**  $j$  to  $\Omega_1(k);$  /\*find a faulty word, and record its address in the fault register \*/

$k=k+1;$

**end\_do**

**end\_do**

**write** "EOF" to  $\Omega_1(k);$  /\*channel  $i$  finishes scanning \*/

**write** "Ready" to  $X_V;$

**while** ( $X_E \neq \text{zero}$ ) **write** "Ready" to  $X_V;$  /\*wait until all channels finish scanning\*/

**for** ( $n=A$  to  $D$ ) /\* re-align faulty words one by one, starting from channel  $A$  to  $D$ \*/

$k=1;$  /\* pointer of channel  $n$  \*/

**while** ( $X_R \neq \text{"EOF"}$ )

**do**

**write**  $\Omega_1(k)$  to  $X_n;$  /\*only channel  $n$  can make a simplex data-exchange  
other channels' **write** commands will be ignored by the system \*/

**read**  $X_R;$  /\*every channel reads the address of the faulty word in channel  $n$ \*/

**if** ( $X_R \neq \text{"EOF"}$ )

**do**

$T = X_R;$  /\* $X_R$  contains the address of the faulty word\*/

**write**  $M(T)$  to  $X_V;$  /\*channels vote on the faulty word\*/

**write**  $X_R$  to  $M(T);$  /\*channels write the voted result back\*/

**end\_do**

$k=k+1;$

**end\_do**

**end**

**Scheme\_1** is more robust than **Scheme\_2**, because in **Scheme\_1** all channels are executing identical instructions in lock step, and any mismatch between channels can be easily detected. Thus, fault-free channels can always complete memory re-alignment without being affected by faulty channels. On the other hand, **Scheme\_2** is faster but more prone



to errors, because the completion of memory re-alignment can be guaranteed only when faulty channels can correctly interact with fault-free channels. For example, if the CPU program counter in one channel stops at a certain point, all the other channels running **Scheme\_2** will be stuck in waiting loops. Although this problem can be easily fixed by adding a time-out to each waiting loop, **Scheme\_2** needs a substantial modification to make it robust.

Both **Scheme\_1** and **Scheme\_2** induce a fixed overhead  $Wt_m$  to scan main memory, where  $t_m$  is the memory cycle time. (Due to its unimportance to the optimization problem to be discussed, this fixed overhead will not be mentioned in the rest of this section.) The performance overhead of **Scheme\_2** is linearly proportional to the total number of faults, whereas **Scheme\_1** may be substantially slower than **Scheme\_2**, i.e., **Scheme\_1** is faster than **Scheme\_2** only when  $g > K + m\frac{W}{K}$ , where  $g$  is the total number of faults, and  $m$  is the number of re-aligned recovery pages, because the value of  $K$  in **Scheme\_1** can be greater than the value of  $g$  in **Scheme\_2**.

The speed of **Scheme\_1** is primarily determined by the size of recovery page and system reliability. Denote the number of recovery pages to be re-aligned by a random variable  $F$ ,  $0 \leq F \leq K$ . Then, the memory re-alignment time is

$$t_{ra} = (K + F\frac{W}{K})t_v,$$

where  $t_v$  is the time to take a vote, i.e., the total time to write  $X_V$ , and read  $X_R$  and  $X_E$ . From Eq. (3.8) it is not difficult to see that the perfect fault detection assumption is reasonable even when the channel failure rate is high and CEMs have only fault detection capability, e.g., even/odd parity codes. When CEMs have only perfect *detection* capability, the probability that  $f$  faulty recovery pages have occurred in the system by time  $t$  is  $P_K(F = f) = \binom{K}{f} R_p^{(K-f)}(t) (1 - R_p(t))^f$ , where  $1 - R_p(t)$  is the probability that one or more of the recovery pages which are in different channels but have the same page number are faulty. Let  $\lambda$  and  $q$  denote the failure rate of a memory word and the number of redundant channels, respectively, then  $R_p(t) = e^{-q\frac{W}{K}\lambda t}$ . Let  $\psi = Wq\lambda t$ , and  $K$  have been

determined, then the conditional probability of  $f$  recovery pages needing to be re-aligned is

$$P_K(t_{ra}) = P(t_{ra} = K + f \frac{W}{K} | \text{memory is faulty}) = \frac{\binom{K}{f} e^{-\psi(1-\frac{f}{K})} (1 - e^{-\frac{\psi}{K}})^f}{1 - e^{-\psi}}. \quad (4.1)$$

The objective of recovery page design is to minimize the re-alignment overhead so that at time  $t$ , the probability of re-alignment requiring more than a time period  $T$  is less than  $\epsilon$ . Therefore, a solution  $K$  is feasible when  $P_K(t_{ra} > T) < \epsilon$ . The optimization problem is essentially a non-linear integer programming problem, and can be stated formally as follows:

$$\begin{aligned} \min \quad & Z(t) = K + F \frac{W}{K} \\ \text{subject to} \quad & K \in \mathbb{I}^+, K < W \\ & P_K(t_{ra} = K + f \frac{W}{K}) = \binom{K}{f} e^{-\psi(1-\frac{f}{K})} (1 - e^{-\frac{\psi}{K}})^f / (1 - e^{-\psi}) \\ & P_K(t_{ra} > T) < \epsilon. \end{aligned}$$

When  $T \geq Wt_v$ , the recovery page design is trivial, because the memory can be easily re-aligned by voting on every word. When  $T < Wt_v$ , and the recovery coverage of CEMs is  $c$ , no solution can be found if the optimal page size based on the given  $c$  is not feasible. Since the coverage of CEMs is very high, the design problem can be focused on *page size optimization*, while the feasibility problem can be easily solved by an exhaustive search. When  $K^* \not\gg 1$ , an exhaustive search for  $K^*$  is the only course to take. On the other hand, when  $K^* \gg 1$ , it will be shown that  $K^*$  can be found through a conventional continuous variable optimization technique.

**Lemma 1:** Given  $\psi$  and  $K$ ,  $P_K(F = f)$ , the probability of  $f$  faults simultaneously occurring to the system, is a monotonically decreasing function of  $f$  when  $\frac{K-f}{f+1}(e^{\frac{\psi}{K}} - 1) < 1$ ,  $1 < f \leq K$ . The sufficient condition for  $P_K(F = f)$  to be a monotonically decreasing function of  $f$  is  $(e^{\frac{\psi}{K}} - 1) < \frac{2}{K-1}$ ,  $K > 1$ .

**Proof:** Since  $P_K(F = f) \geq 0$ ,  $P_K(F = f)$  is monotonically decreasing if  $\frac{P_K(F=f+1)}{P_K(F=f)} < 1$ ,  $\forall f$ . Using Eq. (4.1), we have  $\frac{P_K(F=f+1)}{P_K(F=f)} = \frac{K-f}{f+1}(e^{\frac{\psi}{K}} - 1)$ , or  $P_K(F = f)$  is monotonically decreasing if  $\frac{K-f}{f+1}(e^{\frac{\psi}{K}} - 1) < 1$ . Note that  $0 \leq (e^{\frac{\psi}{K}} - 1) \leq 1$  when  $\frac{\psi}{K} \leq 0.693$ . Since

$\frac{K-f}{f+1} \leq \frac{K-(f+1)}{(f+1)+1}$ , and the maximum value of  $\frac{K-f}{f+1}$  is  $\frac{K-1}{2}$ ,  $K > 1$ , the sufficient condition for the ratio test to hold is  $(e^{\frac{\psi}{K}} - 1) < \frac{2}{K-1}$ ,  $K > 1$ . ■

**Lemma 2:** When the sufficient condition of Lemma 1 holds,  $P(t_{ra} > K + f\frac{W}{K}) < P_K(F = f) \frac{1-\mu_f^{K-f}}{1-\mu_f}$ , where  $\mu_f = \frac{K-f}{f+1}(e^{\frac{\psi}{K}} - 1)$ .

**Proof:** When the sufficient condition of Lemma 1 holds,  $P_K(F = f+1)/P_K(F = f) < \mu_f$ , and  $\mu_f < 1$ . Since  $\mu_f < \mu_{f+1}, \forall f$ , we have  $\sum_{i=f}^K P_K(F = i) < P_K(F = f)(1 + \mu_f + \mu_f^2 \cdots + \mu_f^{K-f})$ , or  $P(t_{ra} > K + f\frac{W}{K}) < P_K(F = f) \frac{1-\mu_f^{K-f+1}}{1-\mu_f}$ . ■

Note that  $\psi \ll K$  holds for most realistic parameter values. When Lemma 1 holds, and  $K$  and  $\epsilon$  are given,  $f_K = \inf f_i$ , such that  $P(t_{ra} > K + f_i\frac{W}{K}) < \epsilon, \forall i$ , can be determined by applying Lemma 2 repeatedly. That is, when the main memory consists of  $K$  pages, with a probability of greater than  $1 - \epsilon$  that the number of faulty pages is less than or equal to  $f_K$ . The next lemma states a key condition that can greatly simplify the optimization algorithm.

**Lemma 3:** If  $K_1(K_2) \gg 1$ ,  $K_1(K_2) \gg \psi$ , and  $K_1(K_2) \gg f$ , then  $P_{K_1}(F = f) \approx P_{K_2}(F = f)$ , where  $P_{K_i}(f)$  is the probability of  $F = f$  when the number of recovery pages is  $K_i$ .

**Proof:**  $P_K(F = f) = \binom{K}{f} e^{-\psi(1-\frac{f}{K})} (1 - e^{-\frac{\psi}{K}})^f$ . When  $K \gg f$ ,  $\binom{K}{f} \approx \frac{K^f}{f!}$ , and  $e^{-\psi(1-\frac{f}{K})} \approx e^{-\psi}$ . Furthermore, when  $K \gg \psi$ , we get  $1 - e^{-\frac{\psi}{K}} \approx 1 - (1 - \frac{\psi}{K}) = \frac{\psi}{K}$ . Combining the above expressions leads to  $P_K(F = f) \approx \frac{K^f}{f!} e^{-\psi} (\frac{\psi}{K})^f$ , or  $P_K(F = f) \approx e^{-\psi} \frac{\psi^f}{f!}$ . That is,  $P_K(F = f)$  is predominately determined by  $f$ , and is insensitive to  $K$ . Thus,  $P_{K_1}(F = f) \approx P_{K_2}(F = f)$  holds. ■

Lemma 3 is valid for a very broad range of  $K$  values, and when  $K_1, K_2 \gg 1$ ,  $P_{K_i}(F = f)$ 's are very close to each other. When Lemma 1 holds,  $P_{K_1}(F = f_1) < P_{K_2}(F = f_2)$ , where  $f_1 > f_2$ .  $P_K(F = f)$ 's with different  $K$  values are plotted in Fig. 7. In these examples, the system has  $W = 4M$  words of memory,  $q = 4$  channels,  $\lambda = 0.75$  byte/ $10^9$  hours, and

$t = 150$  hours. Thus,  $\psi = \lambda tqW = 1.8$ , Thus,  $|P_{500}(F = 1) - P_{13500}(F = 1)| < 0.05$ , and  $|P_{500}(F = 3) - P_{13500}(F = 3)| < 0.001$ . Denoting the optimal value of  $K$  by  $K^*$ , the most desirable property of Lemma 3 is that when  $K^* \gg 1$ , we get  $f_K \approx f_{K^*}$ , and thus,  $K^*$  can be found by the following Theorem.

**Theorem 1:** When  $K^* \gg 1$ ,  $K^* \approx \sqrt{f_K W}$ , where  $K$  is the number of memory pages,  $1 \ll K < W$ .

**Proof:** From Lemma 3, we get  $P_{K_1}(F = f) \approx P_{K_2}(F = f), \forall f$ . Thus, when  $K^* \gg 1$ , we have  $f_K \approx f_{K^*}$ , or  $f_{K^*}$  can be found by applying Lemma 2 to an arbitrary  $K$  such that  $P(f > f_K) < \epsilon$ . Clearly, for a given  $\epsilon$ ,  $f_K \approx \hat{f}, \forall K \gg 1$ , where  $\hat{f}$  is some constant. The cost function  $Z(t)$  to be minimized can be expressed as  $\min_{K \gg 1} (K + \hat{f} \frac{W}{K})$ . Since the objective function is convex when  $K$  is continuous, the optimal solution of real-valued  $K$ 's is  $K' = \sqrt{f_K W}$ . Then,  $K^*$  can be found by an exhaustive search in  $[K' - \delta, K' + \delta]$ , where  $\delta$  is some constant yet to be found. ■

Given the probability,  $\epsilon$ , of the system failing to complete the re-alignment before time  $t_r$ , we want to find the number of faulty pages,  $f_K$ , that can be re-aligned with an approximate value of  $K$ . Since  $f_K \approx f_{K^*}$  when  $K^*$  is large, we derive a near-optimal page size  $K'$ , and  $K^*$  is then derived from  $K'$  by an exhaustive search. An example cost function  $Z(t)$  is plotted in Fig. 8. The curve shown in Fig. 8 is  $K + f_K \lfloor \frac{W}{K} \rfloor$ . The integral constraints on  $K$  and  $\lfloor \frac{W}{K} \rfloor$  cause the sawtooth curve in  $[K - \Delta K, K + \Delta K]$ , but have only a small effect on the global curve shape. In this example,  $\epsilon = 10^{-5}$ ,  $\psi = 1.8$ , and thus  $f_K = 10$ . Thus,  $K' = \sqrt{10 \times 4 \times 10^6} = 6324.5$ . (When  $f_K = 40$ , a similar requirement can be met, but the number of pages is nearly doubled. This is because we want to reduce the page size (and thus increase the number of pages) to reduce the total page re-alignment time.) Through an exhaustive search, it is found that there are multiple optimal solutions, and the one closest to  $K'$  is 6320. The discrepancy between the result obtained from Theorem 1 and the exact solution is due to the integral constraints on  $K$  and  $\frac{W}{K}$ . Thus, having found  $K'$ , the optimal solution can be easily found by using  $K^* = \min K, \lfloor \frac{W}{K} \rfloor = \lfloor \frac{W}{K'} \rfloor$ . However, from a practical

viewpoint, the difference between  $K'$  and  $K^*$  is less than 0.1 %, and thus, it is reasonable to use  $\lfloor K' \rfloor$  as an optimal solution.

From the above example, we can see that even when CEMs have fault detection capability only, the performance of **Scheme\_1** is nearly thousand times better than voting on every word. The performance will be further improved if CEMs also have fault recovery capabilities. Using the example shown in Fig. 8,  $W = 4 \times 10^6$  and  $1 - C_C \approx 2 \times 10^{-7}$  for SEC/DED codes, we get  $f'_K = 1$ , and  $K^* \approx 2000$ , when  $\epsilon$  remains the same ( $10^{-5}$ ).

Cost functions for systems with and without SEC/DED codes are plotted in Fig. 9. When the memory access time is 500 nanoseconds, it takes 2 seconds for a channel to scan main memory. The total re-alignment times for systems without CEMs is 11 seconds. On the other hand, when a fault occurs in a QMR\_CEM system, with a probability greater than  $1 - 10^{-5}$ , it will take less than 2.045 seconds to complete memory re-alignment.

## 5 Conclusion

The reliability of redundant computing systems used for ALS is analyzed and some design issues are discussed. The concept of access set is used for the analysis of near-coincident channel faults leading to system crash. When fault arrivals are independent and the system is free of error propagation and latent faults, the probability of system crash due to multiple channel faults is dictated primarily by component failure rates. It is shown that with the state-of-the-art technology, the probability of system crash due to near-coincident channel faults is insignificant even when the system size is fairly large.

The case study of ALS has shown that the chief cause of unreliability in large redundant systems is the depletion of hardware resources (as a result of component failures), especially when the system has a long mission time. It is worth mentioning that our evaluation of the effectiveness of CEMs in the ALS is very conservative, because all transient faults are assumed to be recoverable by NMR\_CEM and conventional NMR systems. Since transient



faults are typically 10 times more frequent than permanent faults [8, 9], the reliability improvement by using CEMs would be even greater when conventional systems do not have perfect recovery capability for transient faults.

Although emerging new technologies continue to improve hardware reliability and performance, they also stimulate new applications which require higher reliability and computing power. Thus, as main memory is the most vulnerable system component for the current technology, it is expected to be the reliability bottleneck in future computing systems. Fortunately, the design of CEMs for main memory is very simple, and very high fault coverage can be achieved with low overhead. For the example discussed in this report, about 22% of the memory overhead was induced for each channel to embed SEC/DED codes into its main memory. By contrast, adding channels will increase overheads substantially more in the power, physical size and channel synchronization of the system. Thus, embedding SEC/DED codes into main memory is a much more cost-effective method to prolong the resource depletion time than adding more channels to the system.

Large main memory coupled with slack voters makes memory re-alignment very time-consuming. Thus, memory re-alignment in a large system should be avoided whenever possible. It is shown in this report that CEMs can dramatically reduce the need of memory re-alignment, and can speed up the re-alignment process substantially.

Another serious threat to memory re-alignment is the propagation of errors. If error propagation is not effectively prevented, the number of contaminated pages will increase quickly, and thus, the number of pages needing to be re-aligned will increase quickly. Error propagation can be prevented only when the system has very good error detection capability. This is a matter of our future research.

## Acknowledgement

The authors would like to thank F. Pitts, A. White, and R. Butler of the NASA Langley Research Center for their support and comments on an early draft of this report.

## References

- [1] T. B. Smith and J. H. Lala, "Development and evaluation of a fault-tolerant multiprocessor (FTMP) computer Volume I: FTMP principles of operation," NASA Contractor Report 166071, May 1983.
- [2] Charles S. Draper Lab., *Advanced Information Processing System Proof-of-Concept System Fault Tolerant Processor Requirement/Design Specification*, Charles Draper Laboratory Technical Report CSDL-AIPS-84-161, June 3, 1985.
- [3] S. J. Adams, "Hardware assisted recovery from transient errors in redundant processing systems," in *Proc. of 19-th IEEE Fault-Tolerant Computing Symposium*, pp. 512-519, 1989.
- [4] J.-C. Liu and K. G. Shin, "A RAM architecture for concurrent access and on-chip testing," *IEEE Trans. on Computers*, vol. C-40, no. 10, pp. 1153-1159, Oct. 1991.
- [5] R. E. Harper, L. S. Alger, and J. H. Lala, "Advanced Information Processing System: Design and Validation Knowledge Base," NASA Contractor Report 187544, September 1991.
- [6] B. Bose, "Byte unidirectional error correcting codes," processing systems," in *Proc. of 19-th IEEE Fault-Tolerant Computing Symposium*, pp. 222-228, 1989.
- [7] V. 1.0, *METASAN User's Document*, Industrial Technology Institute, P.O. Box 1485, Ann Arbor, MI 48106, Nov. 1987.
- [8] S. R. McConnel, D. P. Siewiorek, and M. M. Tsao, "The measurement and analysis of transient errors in digital systems," in *Proc. of 9-th IEEE Fault-Tolerant Computing Symposium*, pp. 67-70, 1979.
- [9] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Press, 1982.

## Appendix A: List of Symbols

$AS^i, u$	The $i$ -th access set in the system. $AS^i$ is essentially a set of memory words that will be accessed continuously by the CPU (active agent) for a period of time. $u$ is the size of access set.
$f_K, \epsilon$	$K$ is the number of recovery pages in the system. $f_K$ is an upper bound for $f$ , the number of faulty recovery pages, such that $P(t_{ra} > K + f\frac{W}{K}) < \epsilon$ .
$M_k^i$	$M_k^i$ is the length of time that the active agent stays in $AS_k$ during its $i^{th}$ visit to $AS_k$ .
$m$	$m$ is the number of access sets in the system.
$N(t), N^i(t)$	$N(t) = \sum_{i=1}^m N^i(t)$ , where $N^i(t)$ is the number of the agent's visits to $AS^i$ by time $t$ , and $N(t)$ is the total number of visits to access sets by the active agent during $[0, t)$ .
NMR_CEM QMR	NMR_CEM is an N modular redundant system with CEMs embedded into each channel. QMR is a quadruplex modular redundant system.
$P_c(Y_i), P_N(t)$	$P_c(Y_i)$ is the probability of a channel becoming faulty during time interval $Y_i$ . $P_N(t)$ is the probability of system crash caused by multiple channel faults during time interval $[0, t)$ .
$P_K(F = f)$	$P_K(F = f)$ is the probability of $f$ recovery pages becoming faulty when the number of recovery pages is $K$ , and $P_K(t_{ra} = K + f\frac{W}{K})$ is the probability of the re-alignment time $= K + f\frac{W}{K}$ .
$T_i, Y_i$	$T_i$ is the time the $i$ -th vote is held, $Y_i$ is the interval between $T_{i-1}$ and $T_i$ .
$V_j^i, S_k^i$	$V_k^i$ is the event that the active agent makes the $k$ -th visit to $AS^i$ . $S_k^i$ is the moment the event $V_k^i$ begins.
$X_j^i$	$X_j^i$ is a random variable denoting the number of fault occurrences to $AS^i$ during the agent's $j$ -th visit to access sets.

$\lambda_a, \lambda_p, \lambda_i$ $\lambda_m, \lambda_o$	$\lambda_a, \lambda_p, \lambda_i, \lambda_m, \lambda_o$ are the failure rates of an access set, a processor (including control logics), interstage, RAM memory, and ROM memory of each channel in the system, respectively.
$\mu_f, \Phi$	$\mu_f$ is the ratio test of $\frac{P_K(F=f+1)}{P_K(F=f)}$ , $\mu_f = \frac{K-f}{f+1}(e^{\frac{\Phi}{K}} - 1)$ , where $\Phi$ is the product of memory size (words), failure rate of a memory word, number of redundant channels, and the time $t$ .
$\Pi, \Pi_E$ $\Pi_Q$	$\Pi_E$ and $\Pi_Q$ are the environmental and quality factors of a component, respectively. Component failure rate is adjusted by $\lambda' = \Pi_E \times \Pi_Q \times \lambda$ .
$\Omega_i$	$\Omega_i$ is the $i$ -th recovery page.

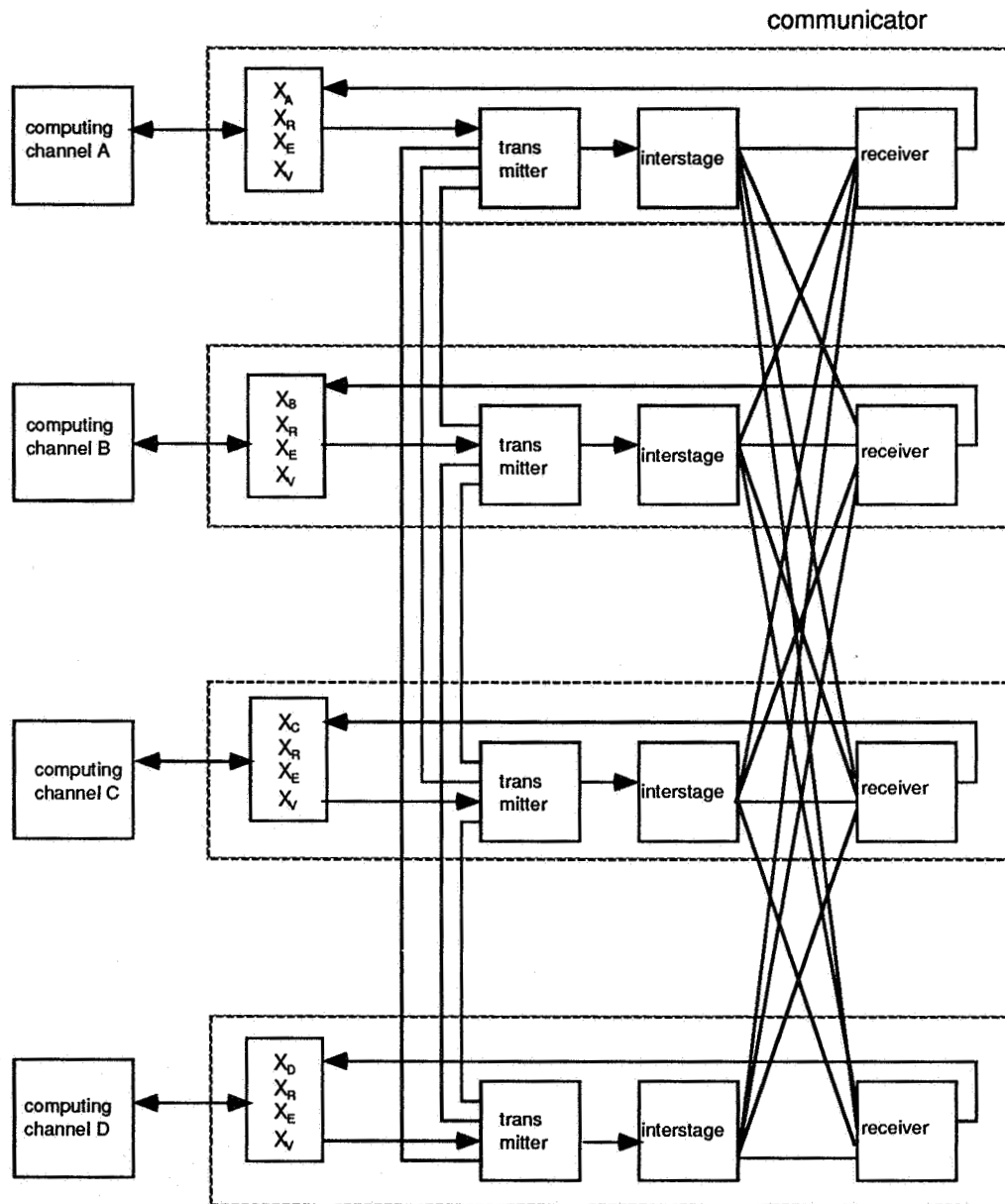


Figure 1: The voting and communication network of computing channels in FTP.



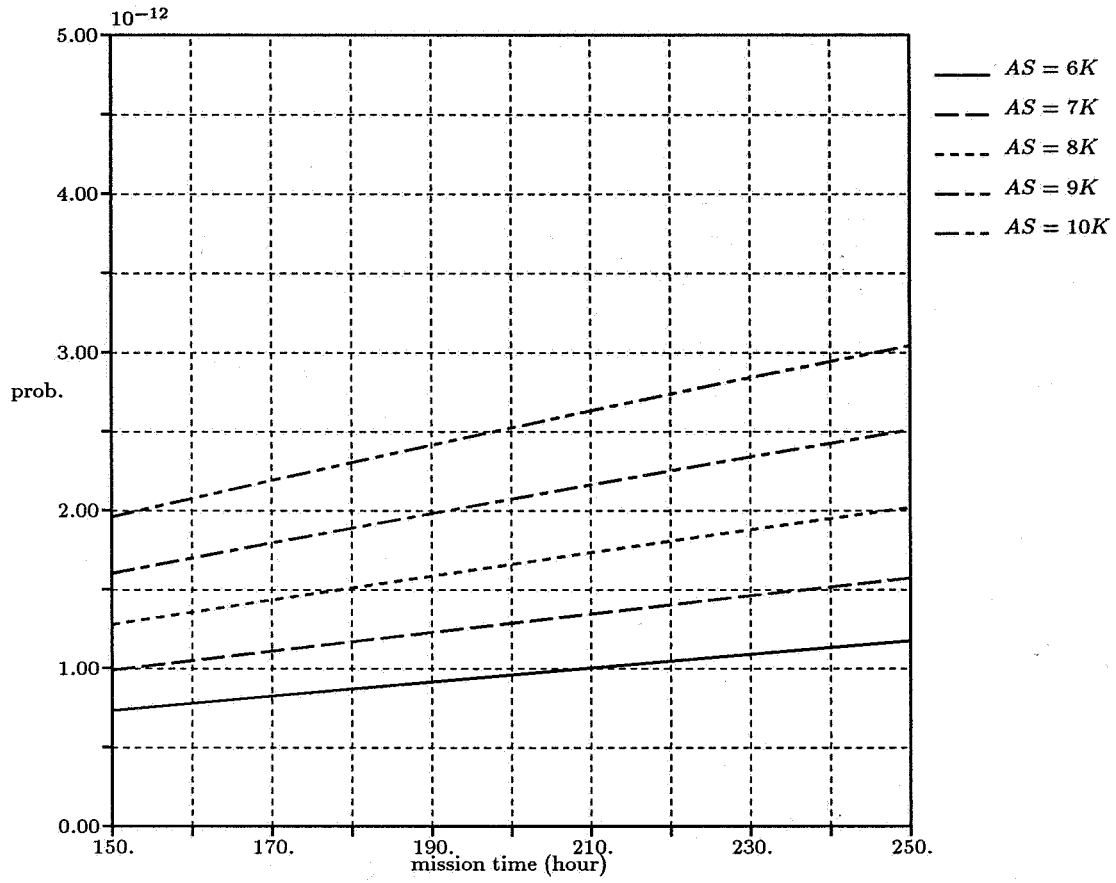


Figure 2: The total probability of nearly-coincident faults with different access set sizes and mission times.

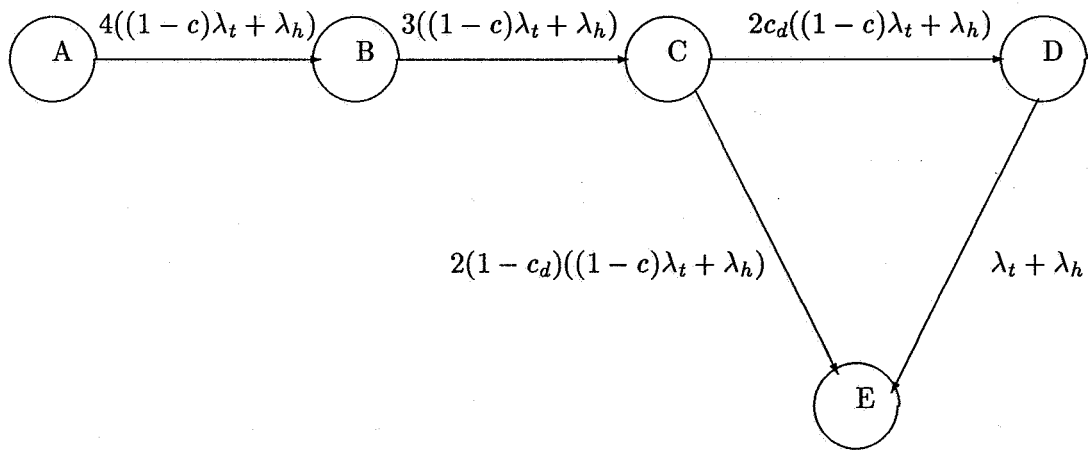


Figure 3: The Markov model of system reliability.

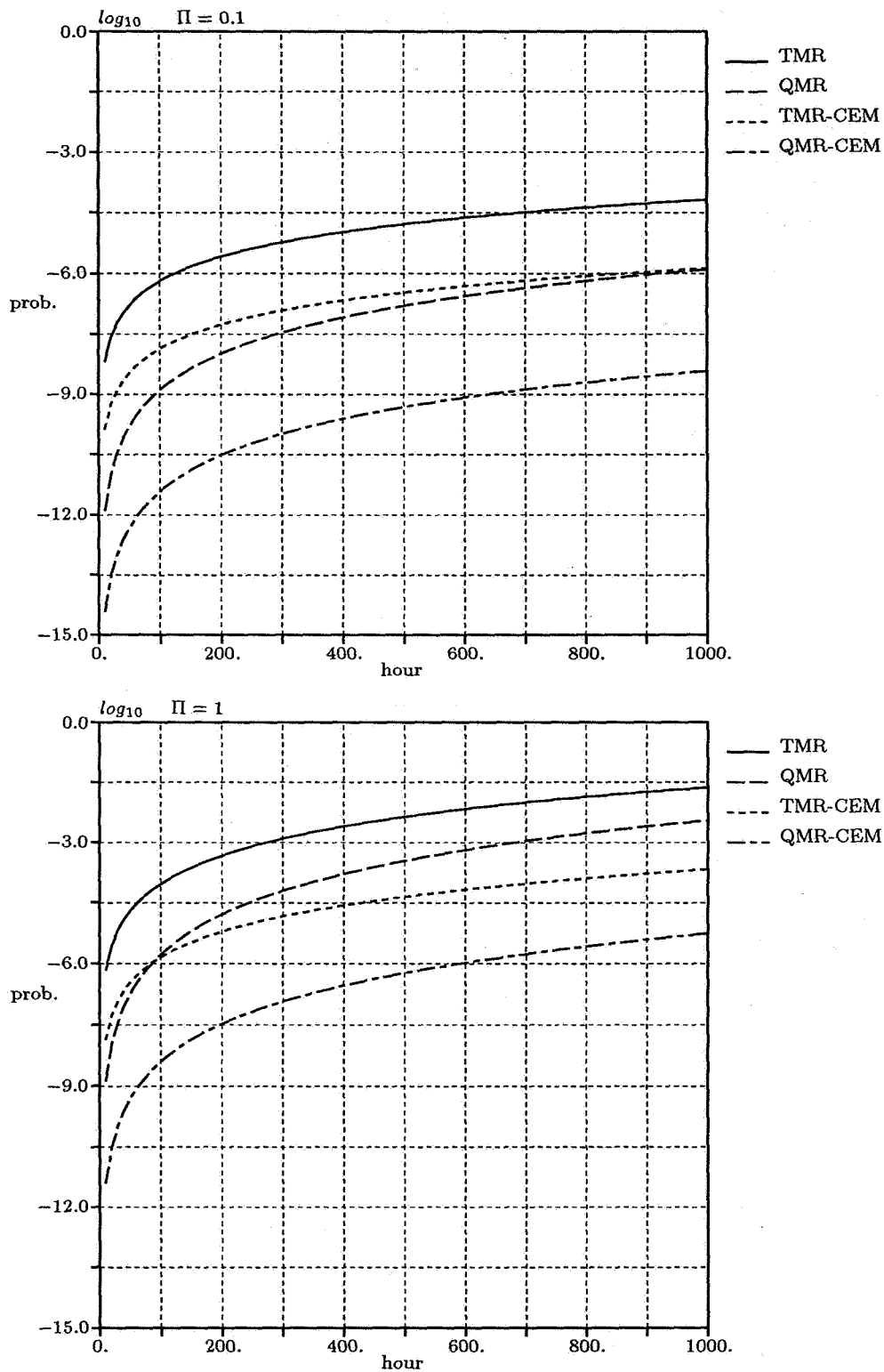


Figure 4: The unreliability of different systems when SEC/DED codes are embedded into RAM when (a)  $\Pi = 0.1$ , and (b)  $\Pi = 1$ .

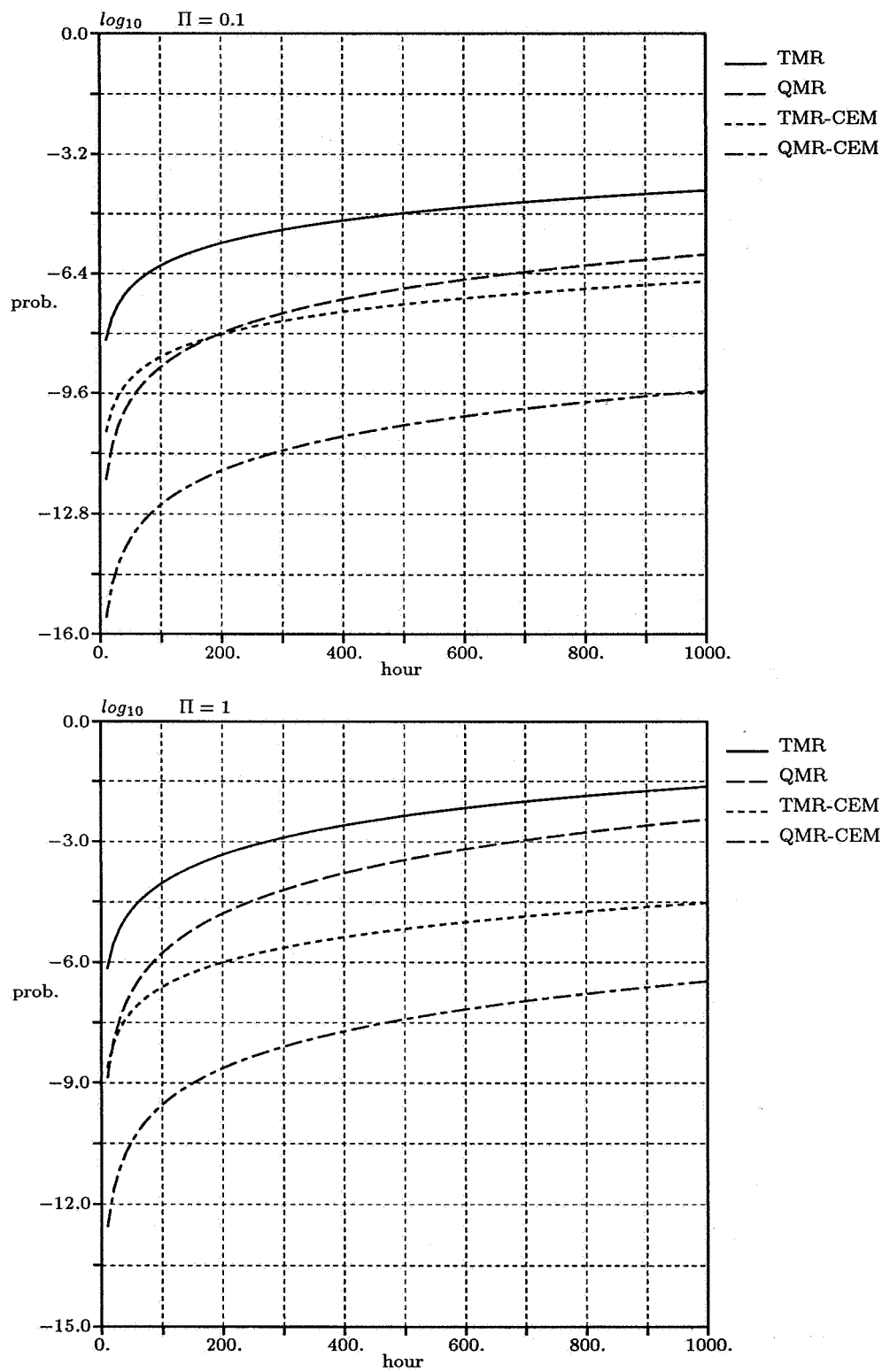


Figure 5: The unreliability of different systems when SEC/DED codes are embedded into ROM and RAM where (a)  $\Pi = 0.1$ , and (b)  $\Pi = 1$ .

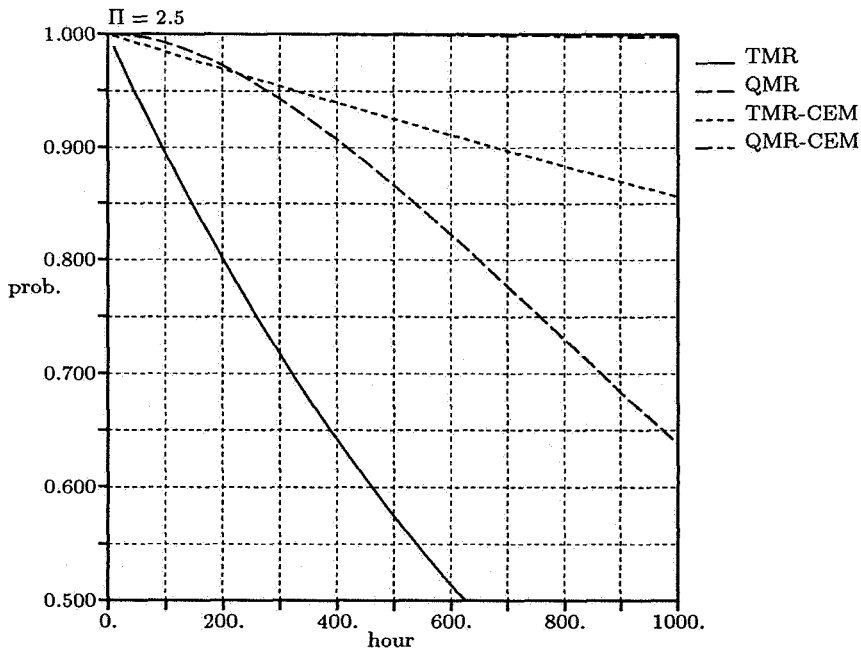
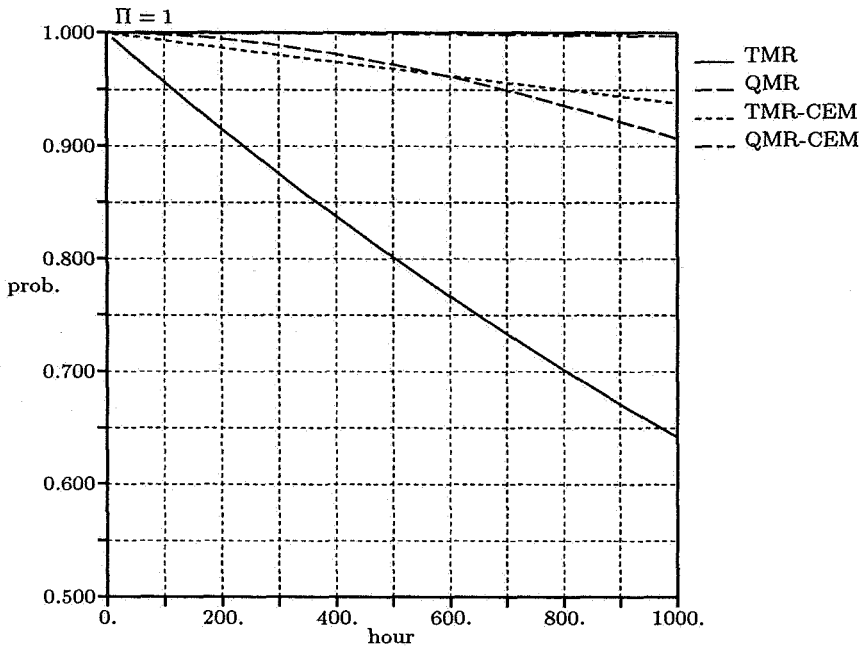


Figure 6: The probability of FTP having fault masking capability before launching when SEC/DED codes are embedded into RAM.

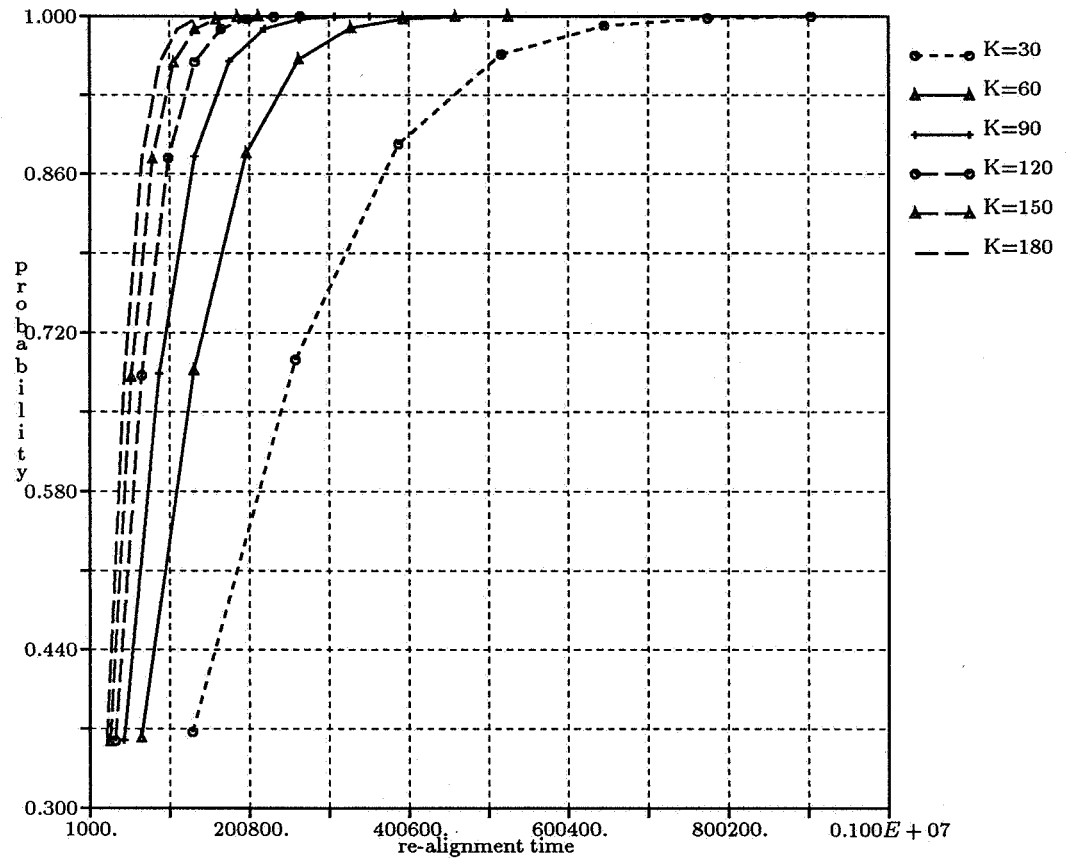


Figure 7: Probability distribution functions of memory re-alignment times when  $t = 200$  hours, the system has 4 channels, each with 4 M words memory.



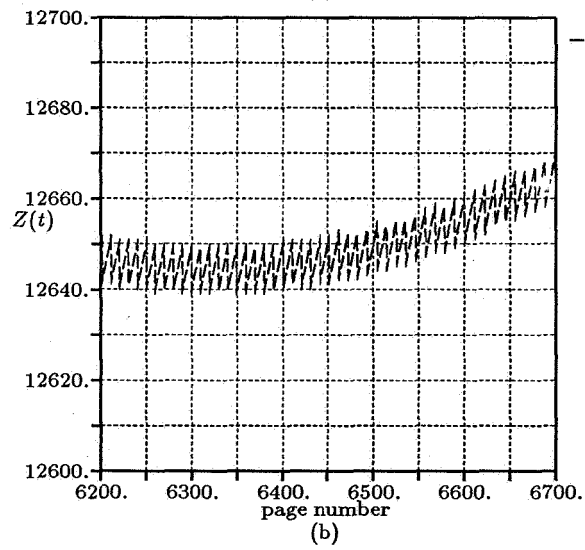
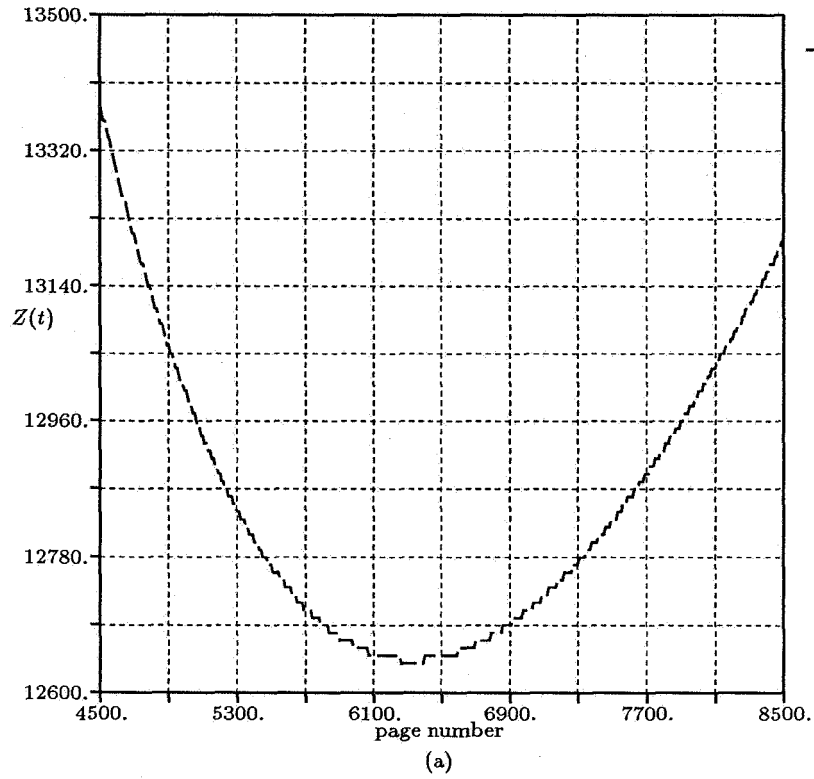


Figure 8: The cost function of a system with perfect detection capability, 4 channels, 4M words,  $t = 150$  hours,  $\lambda = 0.75 \times 10^{-9}$ /hour-word, and  $\epsilon = 10^{-5}$ . (a) The global plot, and (b) a blow up of the cost function around the optimal point.

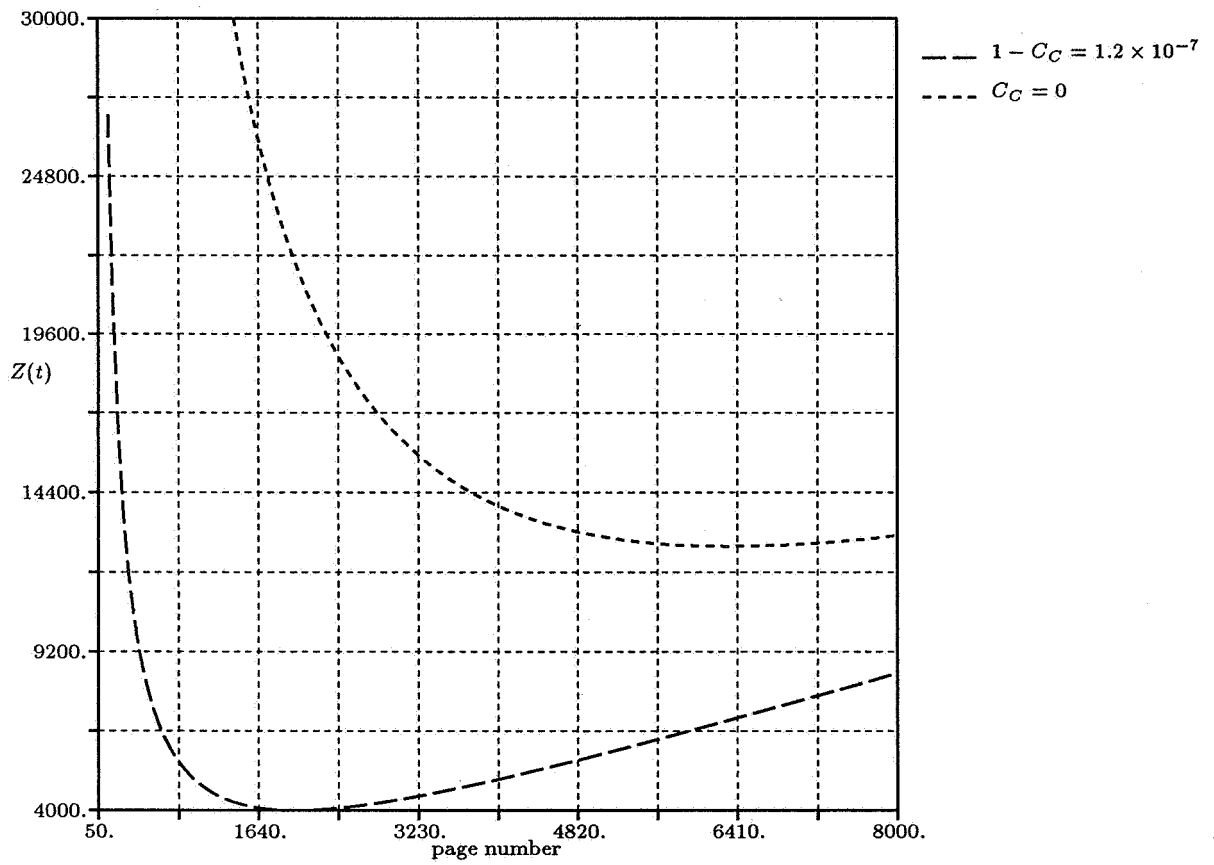


Figure 9: Cost functions of systems with and without recovery capabilities.

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE February 1992	3. REPORT TYPE AND DATES COVERED Contractor Report		
4. TITLE AND SUBTITLE Study on Advanced Information Processing System		5. FUNDING NUMBERS G NAG1-981 WU 946-01-00-61		
6. AUTHOR(S) Kang G. Shin and Jyh-Charn Liu				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The University of Michigan Ann Arbor, MI 48109-2122		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Langley Research Center Hampton, VA 23665-5225		10. SPONSORING/MONITORING AGENCY REPORT NUMBER NASA CR-4428		
11. SUPPLEMENTARY NOTES Langley Technical Monitor: Allan L. White Final Report				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Unclassified - Unlimited  Subject Category 65			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)  Issues related to the reliability of a redundant system with large main memory are addressed in this report. In particular, the Fault-Tolerant Processor (FTP) for Advanced Launch System (ALS) is used as a basis for our presentation. When the system is free of latent faults, the probability of system crash due to nearly-coincident channel faults is shown to be insignificant even when the outputs of computing channels are infrequently voted on. In particular, using channel error maskers (CEMs) is shown to improve reliability more effectively than increasing the number of channels for applications with long mission times.  Even without using a voter, most memory errors can be immediately corrected by CEMs implemented with conventional coding techniques. In addition to their ability to enhance system reliability, CEMs--with a low hardware overhead--can be used to reduce not only the need of memory realignment, but also the time required to realign channel memories in case, albeit rare, such a need arises. Using CEMs, we have developed two schemes, called Scheme_1 and Scheme_2, to solve the memory realignment problem. In both schemes, most errors are corrected by CEMs, and the remaining errors are masked by a voter.				
14. SUBJECT TERMS Access sets; Fault-Tolerant Processor; Fault register; Latent errors; Channel error maskers; Random access memory; Recovery page			15. NUMBER OF PAGES 32	
			16. PRICE CODE A03	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT	